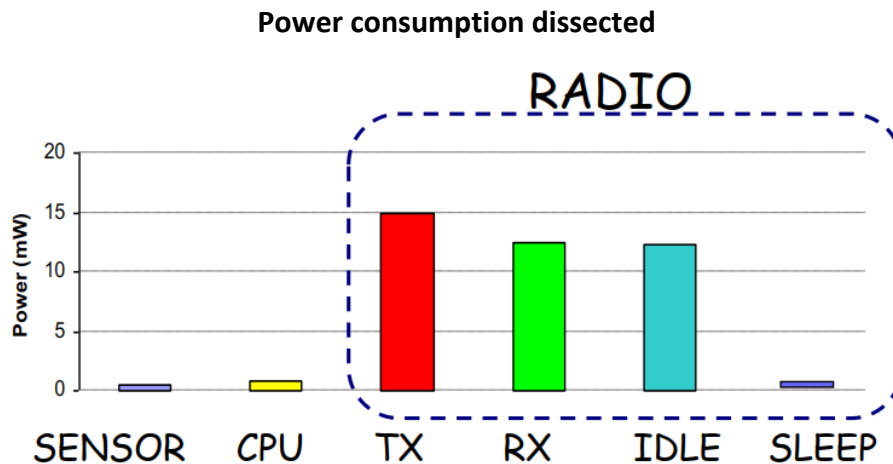


INTERNET OF THINGS SUMMARY

(written by Davide Giannubilo – a.y. 22/23)

3. Energy consumption

If we have IoT devices wireless, we must take care of energy consumption since the sensor has limited power source and its lifetime depends on battery lifetime.

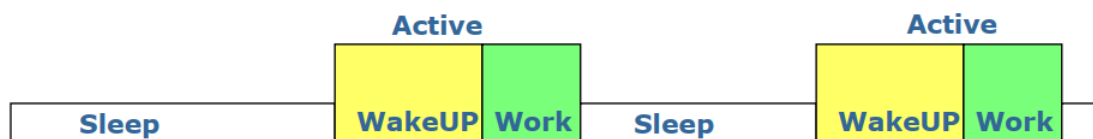


The most power consumption is for *TX* so, for the transmission of a message. Then there is the power consumption for receiving a message (*RX*) and the *IDLE* that is the energy needed to keep the device accessible and ready for processing incoming messages.



Model of operation

The device can be:

- Sleep
- Active
 - Wake-up – the phase in which the device is changing its status
 - Work – the phase in which the device is processing the data



Average power: $P_{ave} = (1 - f_{active}) * P_{sleep} + f_{active} * P_{active}$
($f_{active} \rightarrow duty\ cyle$)

Energy consumption for transmission	
TRANSMISSION	RECEPTION
T_{wu} T_{tx} 	T_{wu} T_{rx} 
$E_{tx} = P_{tx} * (T_{wu} + T_{tx}) + P_o * T_{tx}$	$E_{rx} = P_{rx} * (T_{wu} + T_{rx})$
<p>where</p> <p>P_{tx} is power consumed by transmitter</p> <p>P_{rx} is power consumed by receiver</p> <p>P_o is output power of transmitter</p> <p>T_{tx} time to transmit a packet</p> <p>T_{rx} time to receive a packet</p> <p>T_{wu} is start-up time for transmitter</p>	

What is the consumed energy per bit for transmitting a packet of L [bits]?

Energy spent in emission:	$E_o = P_o * T_L$
Energy spent during wake-up:	$E_{wu} = P_{tx} * T_{wu}$
Energy spent for TX circuitry:	$E_{tx} = P_{tx} * T_L$
Energy per bit:	$(E_{wu} + E_o + E_{tx})/L$

4. introCOMM + APP

4.1 Traffic in IoT and KPIs

- Cyclic Traffic: periodic transfer of sensor measurements and set points (e.g. velocity control loop of an electrical engine requires sending a set point (2 bytes) every 100µs)
- Acyclic Traffic: traffic generated by unpredictable events (e.g. alarm triggered by a failing component of the industrial process)
- Multimedia Traffic: images/video
- Backhand Traffic: aggregated flows

We have some important KPIs:

1. **Throughput (rate)**: number of bits (bytes) per second that can be sent across a link (unit measure: $[bit/s]$ or $[byte/s]$).
2. **Delivery Time**: time necessary to send one service data unit from source to destination (unit measure: $[s]$).

It depends on: transmission time, propagation time, protocol execution time.

$$\begin{aligned} \text{Transmission time} &= L/R \text{ and propagation delay} = d/v \\ T(\text{total}) &= L/R + d/v \end{aligned}$$

(where L is the number of bits, R is the throughput, d is the length of the link between the source S and the destination D)

3. **Minimum Cycle Time (MCT)**: minimum time required to execute a cycle in a control loop (e.g. one controller, 2 sensors; control loop: PLC polls sensor 1, sensor 1 sends back reading, PLC does the same for sensor 2).
4. **Jitter**: precision/reliability in performing periodic operations.

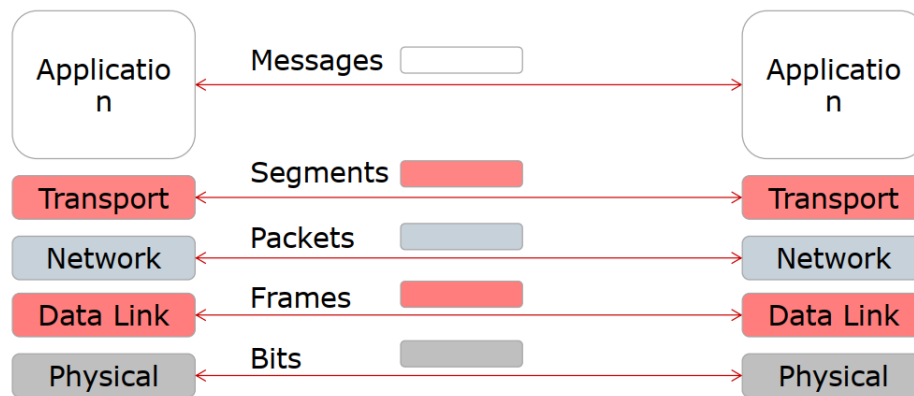
4.2 Networking fundamentals

Communication protocols

Set of rules underpinning the exchange of information between two entities.

- Message format
- Connection set up procedures
- Connection tear down procedures
- Semantics

IoT communication networks protocols exchange packets (like the Internet). They are commonly *layered*. Each layer provides some services/functionalities to upper layers.



4.3 Application Layer protocols for IoT

A program (piece of software) that is in execution at a host and it can communicate with another program through a network (e.g. Safari, Firefox, etc.).

Of course the Internet is populated of various technologies and different languages that have to work together, so they need a mediation.

Based on the type of approach chosen, we have:

- Client/Server protocols, such as
 - HTTP
 - COAP
 - OPC-UA
- Publish/Subscribe protocols, such as
 - MQTT
 - OPC-UA

HTTP (Hyper Text Transfer Protocol)

Web pages composed of resources (HTML, images, applet, ...) and each resource has a Uniform Resource Locator (URL) or Uniform Resource Identifier (URI) and they are accessed synchronously by clients through request/response paradigms. In a word, Representational State Transfer (REST).

How does the HTTP communication work?

Client/Server architecture:

- Client → issues HTTP requests for a specific resource
- Server → answers back with the required resource

Transfer is stateless: no memory maintained at the server.

COAP (COntstrained Application Protocol)

The goal was to enable web-based services in constrained wireless networks with a limited amount of memory and 8-bit micro-controllers.

The result is COAP that is a modified version of HTTP with which shares:

- GET, POST, PUT, DELETE methods
- URI support

- Error codes

and differs from it by:

- Asynchronous transaction model
- Use of UDP instead of TCP at transport layer

The messages exchange between endpoints are characterized by:

- 4 bytes header (shared by request and responses) containing a message ID (16 bits)
- Confirmable messages (reliable messages) that must be acknowledged through ACK or Non-Confirmable Message that must not be acknowledged (non-reliable messages)

Two important elements of the messages:

- ID → defines with a unique number a message and it is necessary in case of CON messages.
- Token → defines uniquely a request.

Imagine having:

- a **CON** request with ID and token
 - we get an **ACK** with same ID and token (if the server answers immediately).
 - we get an **ACK** with same token but different ID (if we receive later the answer from the server).
- a **NON** request with ID and token
 - we do not get an ACK but just a response with same token and different ID.

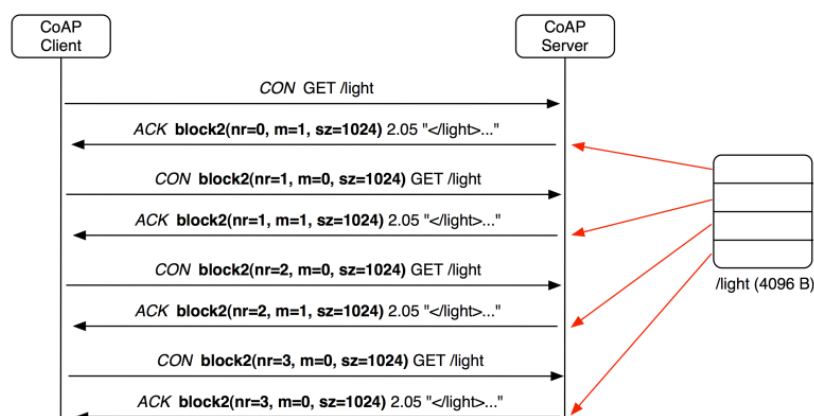
What happens in case of packet loss?

There will be used a “*stop and wait*” approach so, the request will be resent, after a timer-out, in case ACK will not be sent back. The procedure is repeated until an ACK, or an RST is received.

Proxying: it is useful to allow communication between endpoints that are using different protocols.

Caching: useful benefits of caching.

Block transfer: due to the maximum packet dimension of lower layers, it is used to transfer the fragmentation of the packets on the application layer.



(*nr* = incremental block number within original data, *m* = more blocks flag, *sz* = block size)

Discovery: of course, in the COAP protocol, we can use command to discovery resources hosted on a specific server.

```
GET /.well-known/core?optional_query_string
```

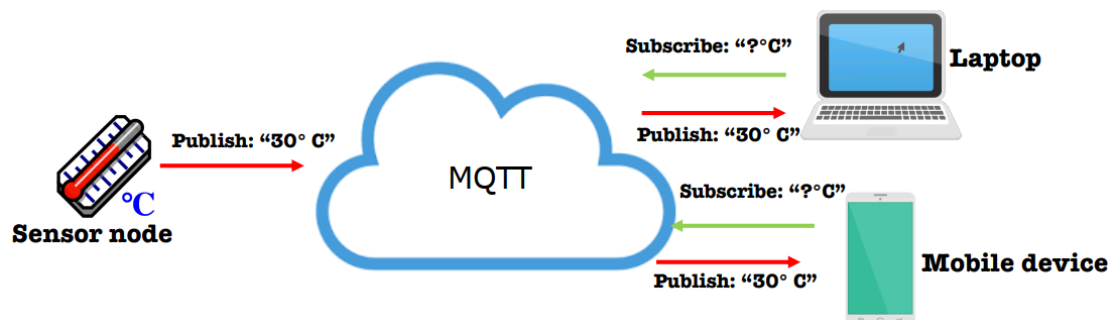
MQTT

MQTT is a Client Server publish/subscribe messaging transport protocol.

Key points:

- Simple to implement (especially at the sensor side)
- QoS Support
- Lightweight and bandwidth efficient
- Data agnostic
- Session awareness

The Publish/Subscribe paradigm is very simple. Clients do not know each other, there is no central server, every client can publish and subscribe, and it is like a *One-to-Many* relationship.



Since we are in a Pub/Sub paradigm, we have also MQTT topics written in the following way:

```
deib/antlab/room5/temperature
```

Wildcards like “+” and “#” (only at the end of the command) are allowed only when subscribing.

How the MQTT connection work?

Very easy, each MQTT client opens one connection to the MQTT broker and sends its message containing:

- clientId
- clientSession
- username (opt)
- password (opt)
- lastWillTopic (opt)
- lastWillQoS (opt)
- lastWillMessage (opt)
- keepalive

After a message is sent, a CONNACK is sent from the MQTT broker to the client containing:

- sessionPresent
- returnCode

- 0: everything ok
- 1: unacceptable version
- 2: ID rejected
- 3: server unavailable
- 4: bad username and password
- 5: unauthorized

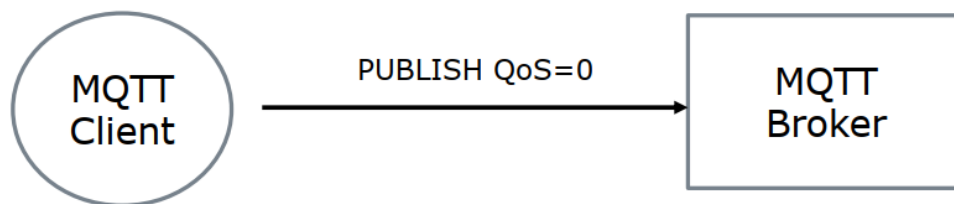
After that a connection is established, the client can start to **publish** its content to all other clients that have been subscribed to the topic to which the content belongs.

Publish message:

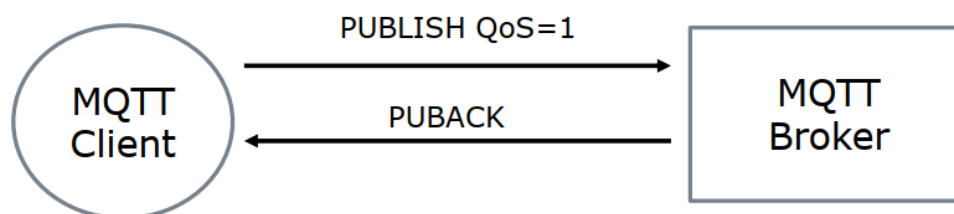
- packetId
- topicName
- QoS
- retainFlag
- Payload
- dupFlag

As mentioned before, MQTT supports **QoS** (Quality of Service) and there three types of it:

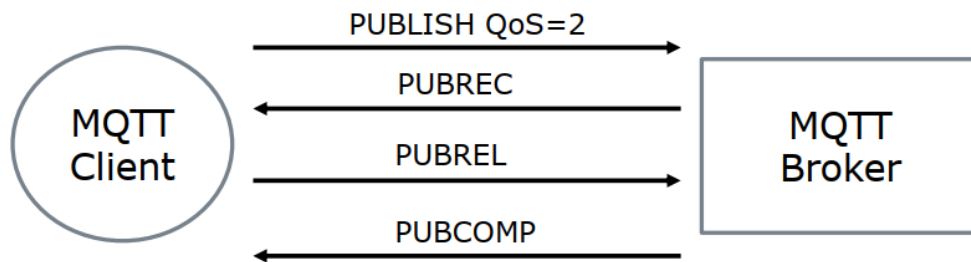
1. **QoS 0**: best effort transfer. Just one message is sent to the broker with no ack sent back.



2. **QoS 1**: the MQTT client stores the message and keeps retransmitting it until it is acknowledged by the MQTT broker (message can be received multiple times). So, PUB message and then ACK from the broker.



3. **QoS 2**: 4 messages between client and broker.
 - 1 – Client sends a message because he wants to publish.
 - 2 – Broker sends back a PUBREC message and stores the packetId to avoid duplicate processing.
 - 3 – Client receives PUBREC and sends PUBREL message to the broker.
 - 4 – Broker receives PUBREL, clears any current state and send back PUBCOMP to the client.



Subscribing works in this way.

- Client sends a SUBSCRIBE message to the broker containing the packetId, the QoS and the topic. Of course, the topics can be more than one.
- Broker receives the subscription message and sends back a SUBACK message with the packetId, and the return code (one for every topic inserted in the SUBSCRIBE message).

Unsubscribing works in the same way of subscribing.

The MQTT protocol also implements the concept of “*persistent session*”. In fact, in default operation mode, when the client disconnects, all the client-related status at the broker is flushed like list of subscriptions or QoS pending messages, but, with this concept, all messages will be stored by the broker and will be available right after the client reconnects.

Retained messages: if the “retainFlag” field is set to True, then the broker saves that message and sends it to all client that will subscribe to the topic, so they will not need to wait for a new message that will be sent.

Last Will and Testament: it is a message that notifies other client about a hard disconnection by a specific client. If a client disconnects gracefully by sending the DISCONNECT message, the LWT will not be sent.

KeepAlive: they are messages sent by the clients to maintain the connection active and to check that the connection is working fine.

MQTT-SN is an extension of MQTT, and it is used for IoT devices. It is characterized by:

- Extended architecture with Gateways and Forwarders
- New gateway discovery procedures (and messages)
- Some messages are more “compressed”
- Extended keepalive procedures to support sleeping clients

6. Wired connectivity

There are three ways to connect IoT devices:

1. Fieldbus
2. Ethernet
3. Wireless

6.1 Fieldbus

This technology is used to connect control and field devices like sensors or actuators. It permits the exchange of short and frequently messages, so it useful when we have tight requirements in terms of delay.

With this technology, we have to share a single communication medium (in our case *IEEE 802.15.4 frequency channel*), so we need to manage the access to it in the right way, and we have two ways:

- Scheduled access → Transmissions on the channel are sequential with no conflicts, polling schemes and centralized scheduling (e.g. GSM, Bluetooth, Wi-Fi PCF)
- Random access → Transmissions are partially uncoordinated and can overlap (collision). Conflicts are resolved using distributed procedures based on random retransmission delay. (e.g. Wi-Fi DCF, Ethernet)

An example of Fieldbus: CAN bus and Profibus

Basically the connectivity is based on shared physical bus. Everybody receives everything transmitted on the BUS.

In order to manage conflicts, the *Carrier Sensing Multiple Access* is used to, first, check if the bus is free, if it is not, it will retry later.

Keep attention also to the vulnerability period. It is a period in which, if another transmission happens in that time window, the conflict cannot be avoided.

Another technique to manage conflicts is called *Bus arbitration*. In this case we have that:

- each message has a priority (Lower identifier field means higher priority),
- each station monitors its own transmission and the status of the BUS,
- if a transmitting station overhears another transmission on the channel at higher priority, then quits

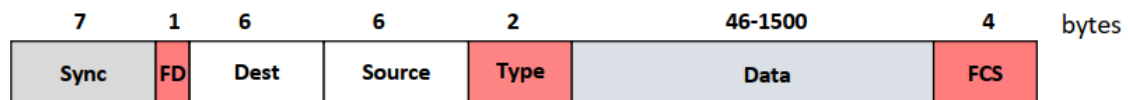
Another technology like the CAN bus is represented by **Profibus**, represented by a master-slave management of communication, in which the master can be changed according to the needs of the network.

6.2 Ethernet

The Ethernet connectivity is based on three main components:

- Physical cables
- Hubs
- Switches

Ethernet frame:



- Synch preamble-Synch (x7 10101010)
- Frame Delimiter- FD (10101011)
- 48-bits Addresses
- Type: multiplexing field (e.g., IP has Type=0800)
- Data field
- Frame Check Sequence – FCS for error checking

An important address is the MAC address that identifies a unique device in the network. It has:

- first 3 bytes set the manufacturer
- last 3 bytes identify the interface

“All ones” address is used for broadcast.

For the collisions, it uses:

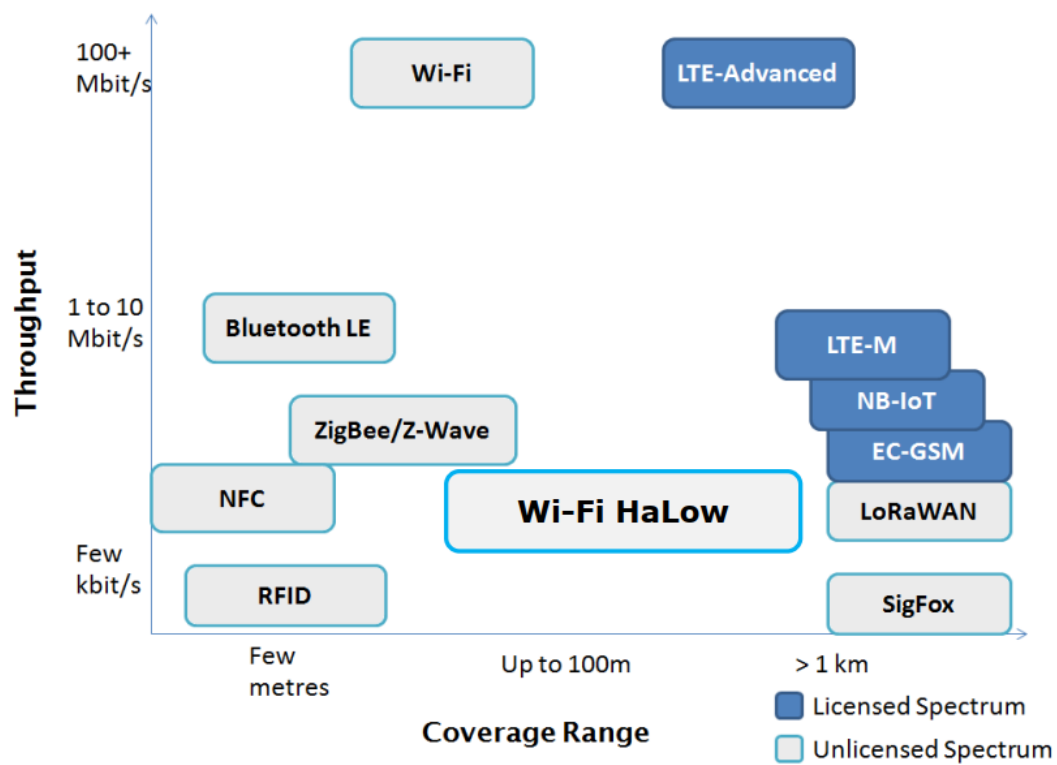
- CSMA/CD, if the bus is free, then the transmission can start otherwise it will retry later.
- Collision Direct, if a collision happens, all the transmissions are aborted. All the frames have the priority.

In today context, the LANs based on Ethernet connectivity are based on *Fully Switched LANs*, in this way we do not have no more collision and no more CSMA/CD.

6.3 Wireless

Although it currently accounts for only 10% of the market, it is a fast-growing industry. It is composed of:

- Mobile Radio Networks
- Cellular IoT Operators
- Capillary Multi-hop Networks



7. Long Range Communication

7.1 Cellular IoT operators

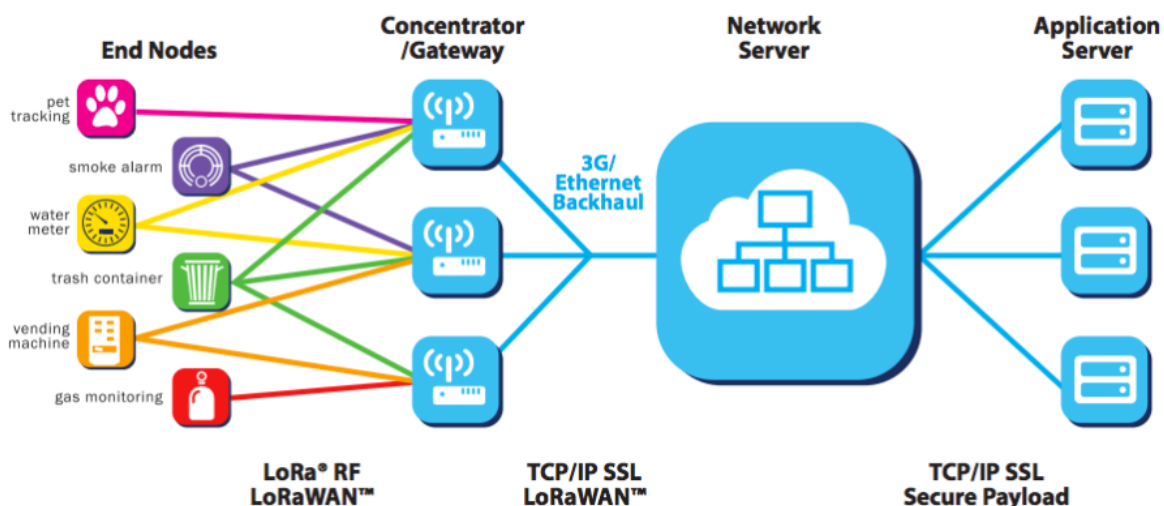
The most important are: SigFox, Ingenu, LoRaWAN and Weightless. They are characterized by:

- Lowest TCO (ownership costs)
- Lowest energy consumption
- Out of the box connectivity (easy to connect)
- Global Reach

7.2 LoRaWAN

This architecture is characterized by:

- Association-less, cellular-like architecture
- End devices: field devices
- Gateways: receive and forward messages from end devices (and network server)
- Network server: where all the intelligence is
 - remove duplicate messages, manages ACKs, manages radio link parameters, etc.



The LoRaWAN protocol stack is composed by:

- First layer that is a physical one that depends on the region we are working on (e.g. US, EU, AS)
- Second layer in which there is the LoRa Modulation that can handle region-specific carrier frequency (this layer is proprietary by SemTech)
- Third layer made by a LoRa MAC address that is open-source and can be 3 classes
 - Class A – baseline
 - Class B – beacon
 - Class C – continuous
- Fourth layer is the application layer

LoRa Modulation

We do not know a lot about this layer since it is proprietary, but we can say that it takes in input some digital data that are converted into a signal specific for the medium used.

One characteristic is that the signal is chipped at a higher data rate and modulated onto a chirp signal.

Chip and bit rates

$$\text{Chip rate } (R_c) = BW \text{ [chips/s]}$$

$$\text{Nominal bit } (R_b) = SF * \frac{4 + CR}{2^{SF}} \frac{4}{BW} \text{ [b/s]}$$

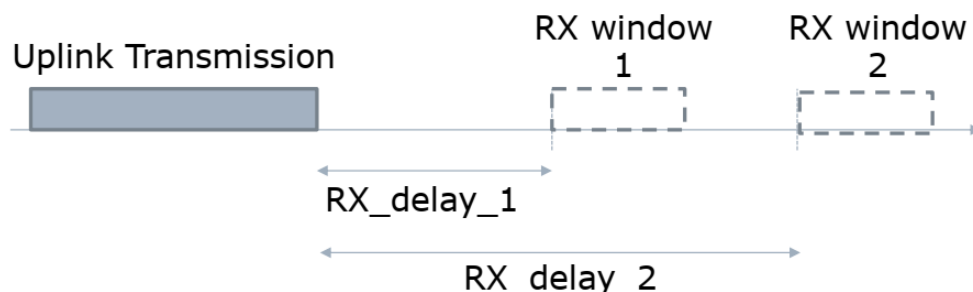
(where BW is the reference bandwidth and SF is the adopted *spreading factor*)

Since SF is a factor chosen by us:

- if it is low, then we have high speed.
- If it is high, then we increase security and reliability.

End devices

- **Class A:** they have the lowest power consumption and are typically the most used type of LoRaWAN device. They have a bi-directional communication method that starts with a device-initiated uplink transmission. They also use ALOHA protocol.



The delays, in EU, are: $RX_delay_1 = 1[s]$ and $RX_delay_2 = RX_delay_1 + 1[s]$.

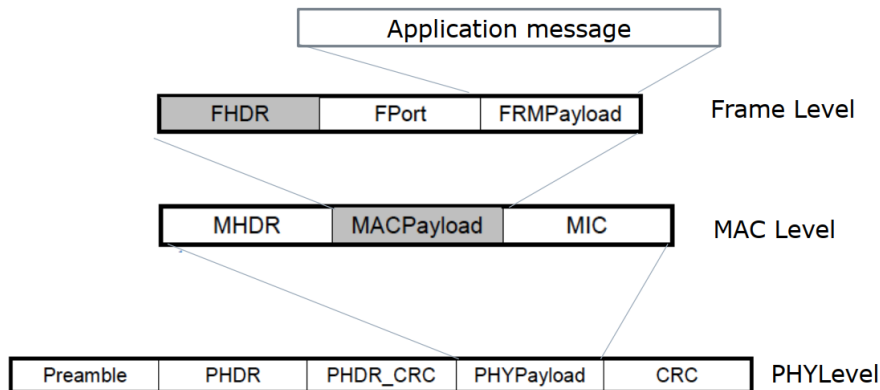
If a *preamble* is detected during one of the receive windows, the radio receiver stays active until the downlink frame is demodulated.

If a frame was detected and subsequently demodulated during the first receive window and the frame was intended for this end-device after address and MIC (message integrity code) checks, the end-device does not open the second receive window.

- **Class B:** are similar to Class A devices but have additional receive windows that are opened at scheduled times, in addition to the receive windows following the uplink transmission. These additional receive windows allow for downlink messages to be sent to the device at specific times, even if it has not recently sent an uplink transmission.
- **Class C:** have the highest power consumption and are designed for applications where low latency is critical. They keep their receive window open continuously, except when they

are transmitting an uplink message, allowing them to receive downlink messages at any time.

LoRaWAN encapsulation



ALOHA protocol

In the ALOHA protocol, each station in the network has an equal chance of transmitting data onto the shared communication channel at any time. When a station has data to transmit, it simply sends the data onto the channel. If two or more stations transmit at the same time, a collision occurs, and the data is lost. In the event of a collision, the stations wait for a random amount of time before trying to transmit again.

ALOHA performance

The probability P_s for a packet transmission to be successful is the probability that no other packet starts transmission in "conflict" period of $2T$.



$$P_s = P(N(t-T, t+T) = 0) = e^{-2G}$$

The throughput is:

$$S = GP_s = Ge^{-2G}$$

LoRaWAN stations may have different $SF \rightarrow$ transmission durations. So imagine having:

- N_1 stations, with arrival frequency λ_1 and transmission duration T_1
- N_2 stations, with arrival frequency λ_2 and transmission duration T_2

The probability that a transmission of type $i(1,2)$ collides, P_i is:

- $P_1 \approx 1 - e^{-(N_1\lambda_1)2T_1} e^{-N_2\lambda_2(T_1+T_2)}$
- $P_2 \approx 1 - e^{-(N_2\lambda_2)2T_2} e^{-N_1\lambda_1(T_1+T_2)}$

The probability that a generic transmission collides is:

$$P = \frac{\lambda_1 N_1}{\lambda_1 N_1 + \lambda_2 N_2} P_1 + \frac{\lambda_2 N_2}{\lambda_1 N_1 + \lambda_2 N_2} P_2$$

In general, if we have n different classes of stations with transmission duration T_i , scale N_i and arrival frequency λ_i , we can generalize the previous expressions as:

$$P_i \approx 1 - \prod_{k=1}^n e^{-(N_k \lambda_k)(T_i + T_k)}$$

$$P = \sum_{i=1}^n \frac{\lambda_i N_i}{\sum_{i=1}^n \lambda_i N_i} P_i$$

7.3 NB-IoT (NarrowBand Internet of Things)

- It is a low-power wide-area networking technology designed to wirelessly connect devices
- Low-cost IoT devices
- Long distances
- It is a part of the 3GPP standard for cellular networks and is also known as LTE-M2.
- It can operate on both licensed and unlicensed spectrum, making it a flexible option for a wide range of applications.

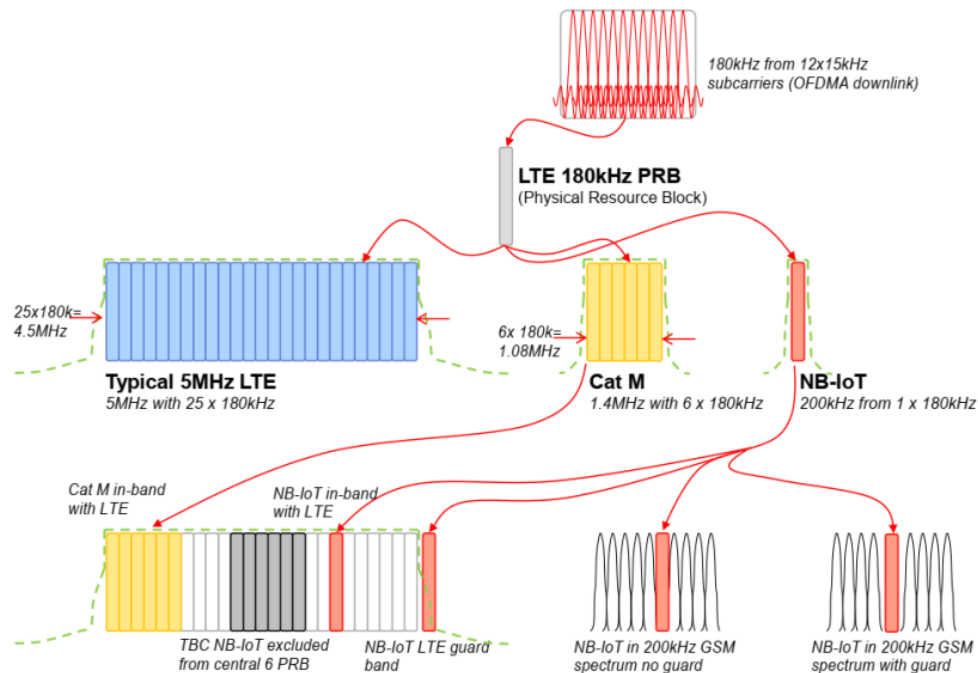
NB-IoT offers several advantages over other wireless networking technologies.

- It provides greater coverage and can penetrate deep inside buildings, making it ideal for applications where devices are deployed in hard-to-reach locations.
- It also supports large numbers of devices per cell, making it suitable for smart city applications, such as smart lighting, waste management, and parking management.
- Support of SMS as a deployment option.
- Reuse of existing mobile network equipment and processes.

NB-IoT offers other features such as enhanced security, extended battery life, and high reliability, making it a popular choice for a range of IoT applications.

Disadvantages to fulfil LPWA requirements:

- No handover support for UEs in the connected state; only cell reselection in the idle state is supported.
- No Intra-RAT interaction with other radio technologies.
- No LTE-WLAN interworking, interference avoidance for in-device coexistence, or measurements to monitor the channel quality.
- Most LTE-Advanced features are not supported.
- No QoS support, as NB-IoT is not used for delay-sensitive data packets.
- Services requiring a guaranteed bit rate, like real-time IMS, are not offered in NB-IoT.



The NB-IoT has channel width under 180KHz.

But where mobile operators place it in the licensed bands that they have?

They three different places:

1. Place these 180KHz in the guard bands of the original LTE channel.
2. Dedicate one of the 25 channels to NB-IoT traffic.
3. Use the guard bands of 2G system.

Guard bands are unused frequencies placed between two operating bands that are intentionally left unused to prevent interference between them and to ensure that the LTE operators can operate efficiently without any degradation in performance.

Power saving mode

It is a special kind of UE status that can minimize the energy consumption that is supposed to be even lower than normal idle mode energy consumption. This is newly added feature in Release 12 and is specified in 3GPP 24.301-5.3.11 *Power saving mode* and 23.682-4.5.4 *UE Power Saving Mode*.

Like power-off, but the UE remains registered with the network and there is no need to re-attach or re-establish PDN connections. A UE in PSM is not immediately reachable for mobile terminating services.